

```

/*****
*****
Nom du fichier:SSI.h
-----
Auteur: Philippe MARIANO
-----
Rôle: fichier d'en-tête de la librairie ssi.lib
-----
Créé le : 10/03/2003
Dernière révision le : 19/12/2009
-----
Cross compilateur utilisé: CodeVisionAVR
-----
Composants cibles : ATMEL 8 bits
-----
Remarques:
Ce fichier est destiné à un projet CVAVR utilisant la carte SSI
Il doit être placé dans le sous répertoire \CVAVR\INC\
Placer le fichier ssi.lib dans le répertoire \CVAVR\LIB\
*****
*****/

#ifndef _SSI_INCLUDED_      // Evite que le fichier ne soit inclus plusieurs fois
#define _SSI_INCLUDED_

// -----
//Interrupt vectors definitions
// -----
#ifndef EXT_INT0
#define EXT_INT0      2
#endif

// -----
// Définition du code des BPs de la carte ATMEL SSI
// -----
#ifndef INC
#define INC          0b00011111      // 0x1f
#endif

```

```

#ifndef DEC
#define DEC          0b00101111      // 0x2f
#endif

#ifndef OK
#define OK           0b00110111      // 0x37
#endif

#ifndef ECHAP
#define ECHAP        0b00111011      // 0x3b
#endif

#ifndef SET
#define SET          0b00111101      // 0x3d
#endif

#ifndef ENTR
#define ENTR         0b00111110      // 0x3e
#endif

// *****
// *****
// Codes de commande de l'afficheur LCD 2x16 à processeur Hitachi type HD44780
// *****
// *****
// Utilisation: A écrire dans le registre de contrôle de l'afficheur
// avec une instruction _lcd_write_data(unsigned char data) précédée d'une instruction
// _lcd_ready(void) de la librairie lcd.h
// *****
// -----
// Initialisation du déplacement du curseur
// -----

#ifndef EntryMode0
#define EntryMode0    0b00000100    // Par défaut déplacement du curseur vers la gauche
#endif

#ifndef EntryMode1
#define EntryMode1    0b00000101    // Par défaut déplacement de l'affichage vers la gauche
#endif

#ifndef EntryMode2
#define EntryMode2    0b00000110    // Par défaut déplacement du curseur vers la droite
#endif

#ifndef EntryMode3

```

```

#define EntryMode3      0b000000111  // Par défaut déplacement de l'affichage vers la droite
#endif
#ifndef clear
#define clear           0b000000001  // Efface l'écran et la DDRAM
#endif
#ifndef home
#define home            0b000000010  // Ramène le curseur au début de la première ligne
#endif
// -----
// Affichage           Code           On/Off           curseur           clignotement
// -----
#ifndef displayOff0
#define displayOff0     0b000001000  // Off           invisible         non
#endif
#ifndef displayOff1
#define displayOff1     0b000001001  // Off           invisible         oui
#endif
#ifndef displayOff2
#define displayOff2     0b000001010  // Off           visible          non
#endif
#ifndef displayOff3
#define displayOff3     0b000001011  // Off           visible          oui
#endif
#ifndef displayOn0
#define displayOn0      0b000001100  // On            invisible         non
#endif
#ifndef displayOn1
#define displayOn1      0b000001101  // On            invisible         oui
#endif
#ifndef displayOn2
#define displayOn2      0b000001110  // On            visible          non
#endif
#ifndef displayOn3
#define displayOn3      0b000001111  // On            visible          oui
#endif
// -----
// Déplacement du curseur ou de l'affichage
// -----
#ifndef curseurG

```

```

#define curseurG      0b00010000  // Déplacement du curseur vers la gauche
#endif
#ifndef curseurD
#define curseurD      0b00010100  // Déplacement du curseur vers la droite
#endif
#ifndef fenetreG
#define fenetreG      0b00011000  // Déplacement de l'affichage vers la gauche
#endif
#ifndef fenetreD
#define fenetreD      0b00011100  // Déplacement de l'affichage vers la droite
#endif
// -----
// -----
// Construction d'un caractère personnalisé. Codes à écrire dans la CGRAM
// aux adresses $40, $48, $50, $58, $60, $68, $70, $78 avec une instruction
// lcd_write_byte(unsigned char adresse, unsigned char data)
// -----
// Exemple de déclaration d'une matrice 5x7, le 8e caractère correspond au curseur
// const unsigned char flecheHaut[8] = {0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04, 0x00};
// 00100
// 01110
// 10101
// 00100
// 00100
// 00100
// 00100
// 00100
// 00000
// *****

// *****
// *****
// Prototype des fonctions générales de la carte SSI
// *****
// *****
// 1 - unsigned char Lire_BP(void);
// 2 - void Affiche_LCD(char *ligne1,char *ligne2);
// 3 - unsigned char eeprom_read(unsigned char addcase, unsigned int adressmap);
// 4 - void eeprom_write(unsigned char addcase, unsigned char adressmap,

```

```
// unsigned char data);
// 5 - void eeprom_page_read(unsigned char addcase, unsigned int adressmap,
// unsigned char *display_buffer);
// 6 - int regle_HTR(unsigned char adresse,unsigned char Heure_ou_Date, char H_J,
// char mn_mois, int sec_annee);
// *****
```

```
#pragma used+
```

```
// -----
```

```
unsigned char Lire_BP(void);
```

```
/* -----
```

Rôle : Lit l'état des BP de la carte ATMEL SSI(PortA) et renvoie un code sur 8 bits.

Commentaires : Dans un premier temps, la fonction renvoie le code de la touche actionnée

(INC = 0x1F, DEC = 0x2f, OK = 0x37, ECHAP = 0x3B, SET = 0x3D, ENTR=0x3E)

puis elle renvoie 0x00 si la touche est maintenue ou 0xFF si la touche est relâchée.

Entrées : aucunes

Sortie : code de la touche ou 0x00 ou 0xFF

```
-----*/
```

```
// -----
```

```
void Affiche_LCD(char *ligne0,char *ligne1);
```

```
/* -----
```

Rôle : Gère l'affichage d'un LCD équipé d'un contrôleur Hitachi (commandé en mode 4 bits)

Commentaires : Envoie le contenu de deux tableaux de 16 caractères sur les lignes 0 et

1 d'un afficheur LCD 2\*16 caractères.

Chaque tableau doit pouvoir contenir 17 valeurs (les 16 caractères à afficher + \0)

Librairie utilisé : lcd.h

Entrées : ligne0, ligne1 = tableaux de 17 caractères

Sorties : Aucune

```
-----*/
```

```
// -----
```

---

```
unsigned char eeprom_read(unsigned char addcase, unsigned int adressmap);
```

```
/* -----
```

Rôle : Lit un octet dans une mémoire série gérée par le protocole I2C

Commentaires : A utiliser avec les mémoires > 256 octets

Librairie utilisée : I2C.h

Entrées :

addcase : Octet de commande du boîtier mémoire. Construit par 1 0 1 0 A2 A1 A0 R/W

remarque (R/W doit être mis à zéro)

adressmap : adresse à atteindre dans le boîtier

Sortie : contenu de l'adresse adressmap du boîtier situé en addcase

```
-----*/
```

```
// -----
```

```
void eeprom_write(unsigned char addcase, unsigned char adressmap, unsigned char data);
```

```
/* -----
```

Rôle : Ecrit un octet dans une mémoire série gérée par le protocole I2C

Commentaires : A utiliser avec les mémoires > 256 octets

Librairie utilisée : I2C.h et delay.h

Entrées :

addcase : adresse du boîtier mémoire construit par 1 0 1 0 A2 A1 A0 R/W

remarque (R/W doit être mis à zéro)

adressmap : adresse à atteindre dans le boîtier

data : octet à écrire à la position adressmap du boîtier situé en addcase

Sorties : aucune

```
-----*/
```

```
// -----
```

```
void eeprom_page_read(unsigned char addcase, unsigned int adressmap, unsigned char *display_buffer);
```

```
/* -----
```

Rôle : Lit 32 octets dans une mémoire série gérée par le protocole I2C

Commentaires : A utiliser avec les mémoires > 256 octets

Librairie utilisée : I2C.h

Entrées :

addcase : adresse du boîtier mémoire construit par 1 0 1 0 A2 A1 A0 R/W  
 remarque (R/W doit être mis à zéro)  
 adressmap : adresse à atteindre dans le boîtier

Sorties : contenu des 32 adresses à partir de adressmap du boîtier situé en addcase

```
-----*/
// -----
int regle_HTR(unsigned char adresse,unsigned char Heure_ou_Date, char H_J, char mn_mois, int sec_annee);
// -----
// Etats du graphe des transitions "REGLAGE"
// -----
#ifndef Regle_Heure_ou_date
#define Regle_Heure_ou_date 0
#endif
#ifndef Regle_min_mois
#define Regle_min_mois      1
#endif
#ifndef Regle_seconde_annee
#define Regle_seconde_annee 2
#endif
#ifndef Mise_a_jour_HTR
#define Mise_a_jour_HTR     3
#endif
/*
-----
```

Rôle : Régle la date et l'heure d'une HTR PCF8583 située à adresse (0 ou 1)

Commentaires :

La fonction règle l'heure si Heure\_ou\_Date = 0 ou de la date si Heure\_ou\_Date = 1.

Les paramètres H\_J, mn\_mois, sec\_annee servent de base aux réglages.

Celui-ci s'effectue avec les BP :

- INC pour augmenter une valeur,

- DEC pour diminuer une valeur,
- OK pour valider la valeur modifiée et passer au réglage suivant
- EChap pour transférer la valeur réglée et sortir de la fonction

Librairie utilisée : PCF8583.h

Entrées :

adresse : adresse du boîtier (0 ou 1 selon A0)

Si Heure\_ou\_Date = 0 (réglage de l'heure) alors -1 < H\_J < 24, -1 < mn\_mois < 60

-1 < sec\_annee < 60

sinon -1 < H\_J < 32, 0 < mn\_mois < 13

1999 < sec\_annee < 2050

Sortie : la fonction renvoie 0

-----\*/

// \*\*\*\*\*

// Prototype des fonctions destinées aux PPE et aux TP

// \*\*\*\*\*

// \*\*\*\*\*

// PPE Acces lycée: D2000\_read, D2000\_write

// PPE Pousse seringue: Menu\_Select\_Seringue, Regdebit

// PPE Doppler: RegVitesse

// \*\*\*\*\*

// \*\*\*\*\*

// Fonctions développées pour le PPE Accès lycée

// -----

unsigned char D2000read(unsigned char adreemap);

/\* -----

Rôle : Lit un octet dans une carte à puce D2000 (gérée par le protocole I2C)

Commentaires : Mémoire eeprom de type PCF8582 (256 octets)

@boîtier figé en 0xa0

Librairie utilisée : I2C.h

Entrées : adressmap = adresse à atteindre dans le boîtier



Sorties : contenu de l'adresse adressmap du boîtier situé en 0xa0

```
-----*/
// -----
void D2000_write(unsigned char adressmap, unsigned char data);
/* -----
```

Rôle : Ecrit un octet dans une carte à puce D2000 (gérée par le protocole I2C)

Commentaires : Mémoire eeprom de type PCF8582 (256 octets)  
Adresse du boîtier figée en 0xa0

Librairie utilisée : I2C.h

Entrées :

adressmap = adresse à atteindre dans le boîtier  
data = caractère à écrire

Sorties : aucunes

```
-----*/

// *****
// Fonctions développées pour le PPE/TP Pousse seringue
//-----
// 1 int Regdebit(unsigned int *dizaines,unsigned int *unites, unsigned int *diziemes);
// 2 unsigned char Select_Seringue(void);
//
// *****
// -----
int Regdebit(unsigned int *dizaines,unsigned int *unites, unsigned int *diziemes);
/* -----
```

Rôle : Renvoie la valeur de la consigne de débit réglée sous la forme dizaine, unité, dixième

Commentaires : fonction écrite pour le pousse seringue

Librairie utilisée : lcd.h et stdio.h

Entrées / sortie : dizaine de ml/h, unité ml/h, dixième ml/h

```
-----*/
// -----
unsigned char Menu_Select_Seringue(void);
```

```

/* -----
Rôle : Renvoie le type de la seringue sélectionnée
      0 : TERUMO 50CC    1 : MONOJECT/ BD 50CC    2 : BD 20CC

Commentaires : fonction écrite pour le pousse seringue

Librairie utilisée : lcd.h et stdio.h

sortie : seringue sélectionnée
-----*/

// *****
// Fonctions développées pour les PPE Doppler
//-----
// 1 int RegVitesse(unsigned int *unites,unsigned int *diziemes, unsigned int *centiemes);
// 2 void ActiveTimer(void);
// 3 void DesactiveTimer(void);
// *****
// -----
int RegVitesse(unsigned int *unites,unsigned int *diziemes, unsigned int *centiemes);
/* -----
Rôle : Renvoie la valeur de la consigne de Vitesse réglée sous la forme unité, dixième, centième
de mm

Commentaires : fonction écrite pour le PPE Doppler

Librairie utilisée : lcd.h et stdio.h

Entrées / sortie : unité mm/s, dixième mm/s, centième de mm/s,
-----*/
// -----
unsigned char Menu_Select_Sens(void);
/* -----
Rôle : Renvoie le sens de déplacement choisi
      0: Rentrer  1: Sortir

Commentaires : fonction écrite pour le PPE Doppler

Librairie utilisée : lcd.h et stdio.h

```

Entrées : rien      Sortie: sens

-----\*/

// -----

void ActiveTimer(void);

/\* -----

Rôle : Autoriser le fonctionnement du timer1

Commentaires : fonction écrite pour le PPE Doppler

Entrées / sortie : rien

-----\*/

// -----

void DesactiveTimer(void);

//-----

/\* -----

Rôle : Inhibe le fonctionnement du timer1

Commentaires : fonction écrite pour le PPE Doppler

Entrées / sortie : rien

-----\*/

#pragma used-

#pragma library ssi.lib

#endif      // termine la directive ifndef